An Autonomic Software Engineering Life Cycle Model

2nd DITA Annual Trainee Conference

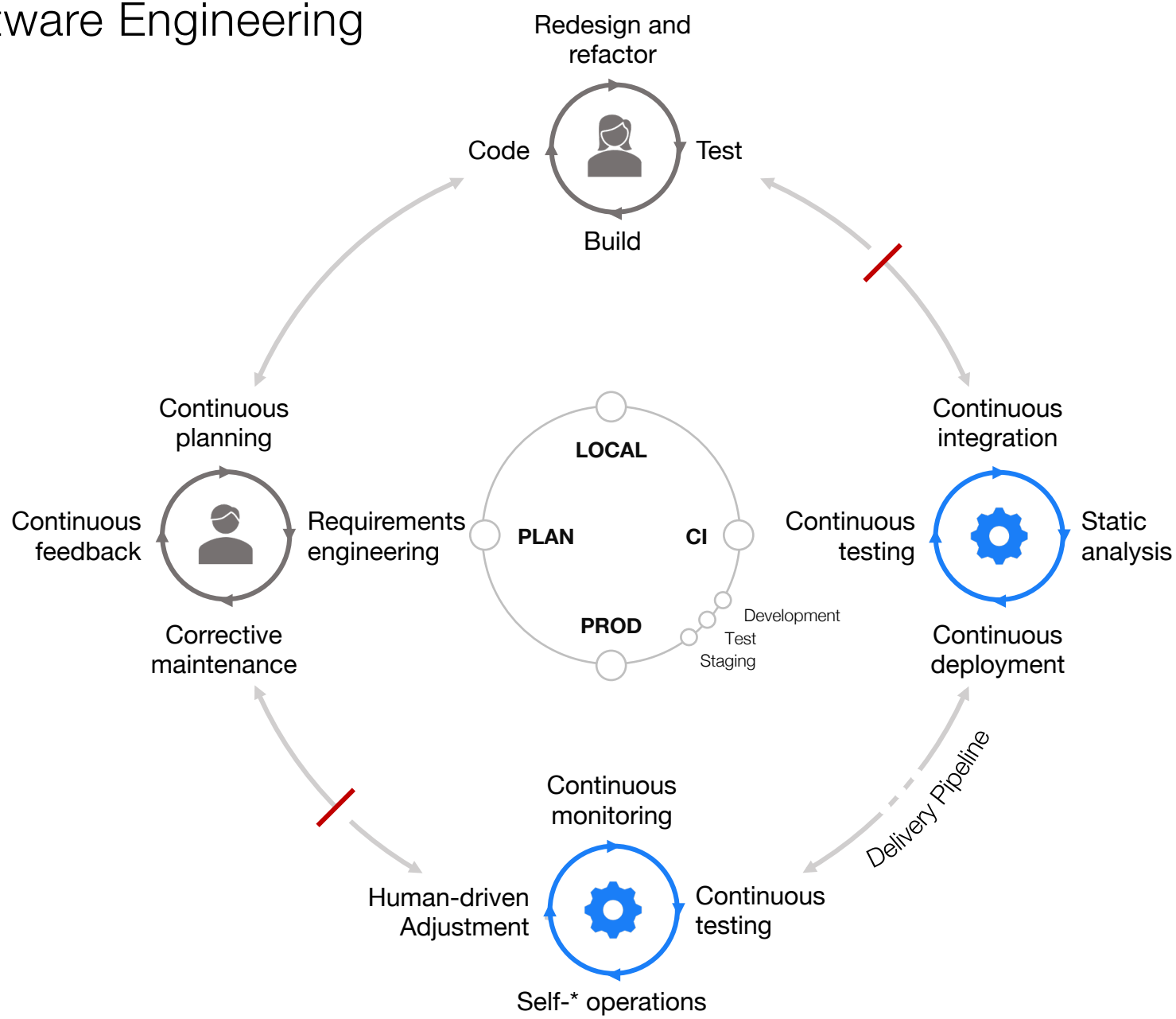
Miguel Jiménez, Felipe Rivera, Norha M. Villegas, Gabriel Tamura, Hausi Müller

July 20th, 2021

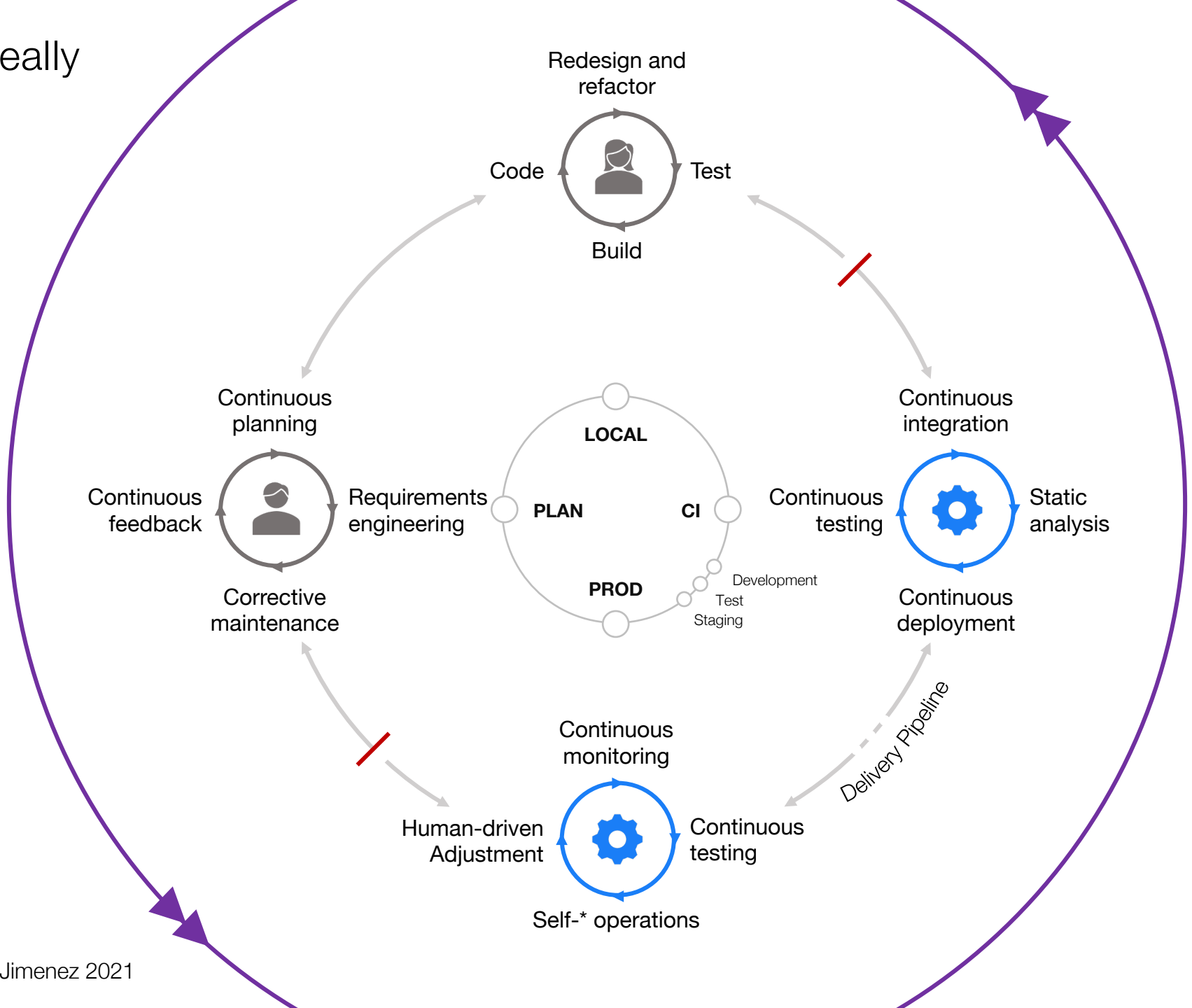
University of Victoria · Rigi Research · Universidad ICESI · DRISO Desarrollo e Ingeniería de Software · IBM · IBM Advanced Studies A system of engagement for academic collaboration · NSERC CRSNG · DITA · CSER

Continuous Software Engineering

- Integration
- Testing
- Deployment
- Delivery
- Design
- Feedback
- Planning
- Monitoring
- …

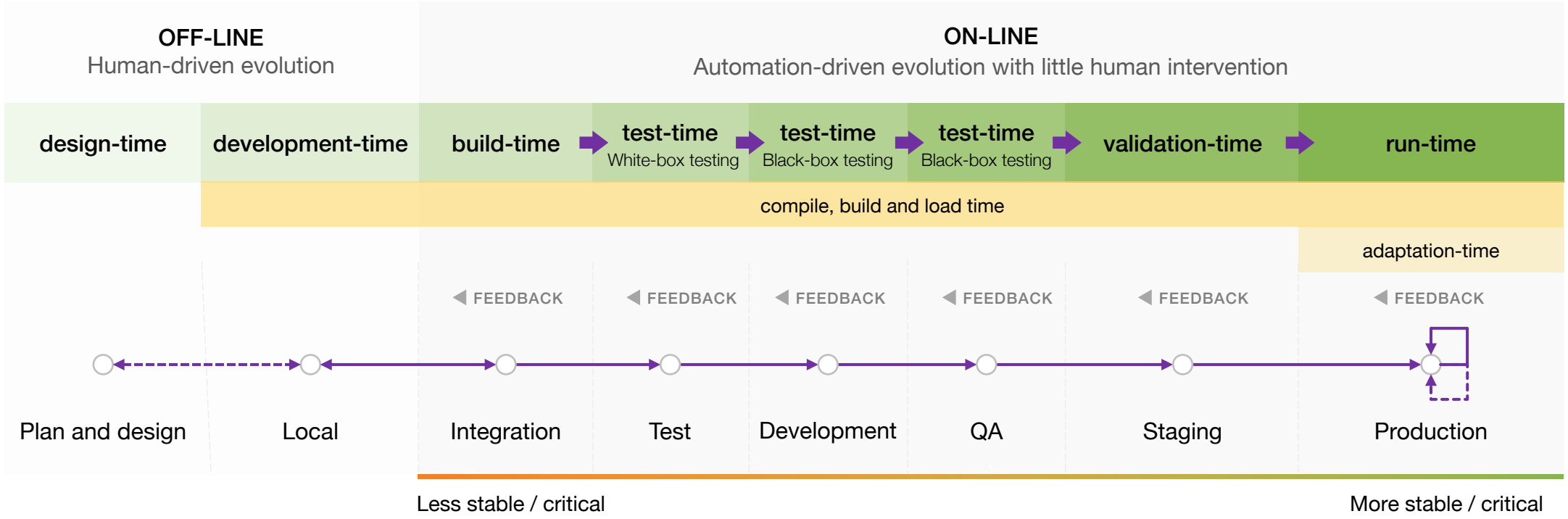


Is the SDLC really continuous?



Off-line and On-line Reconceptualization

A holistic view of the time-of-change timeline today

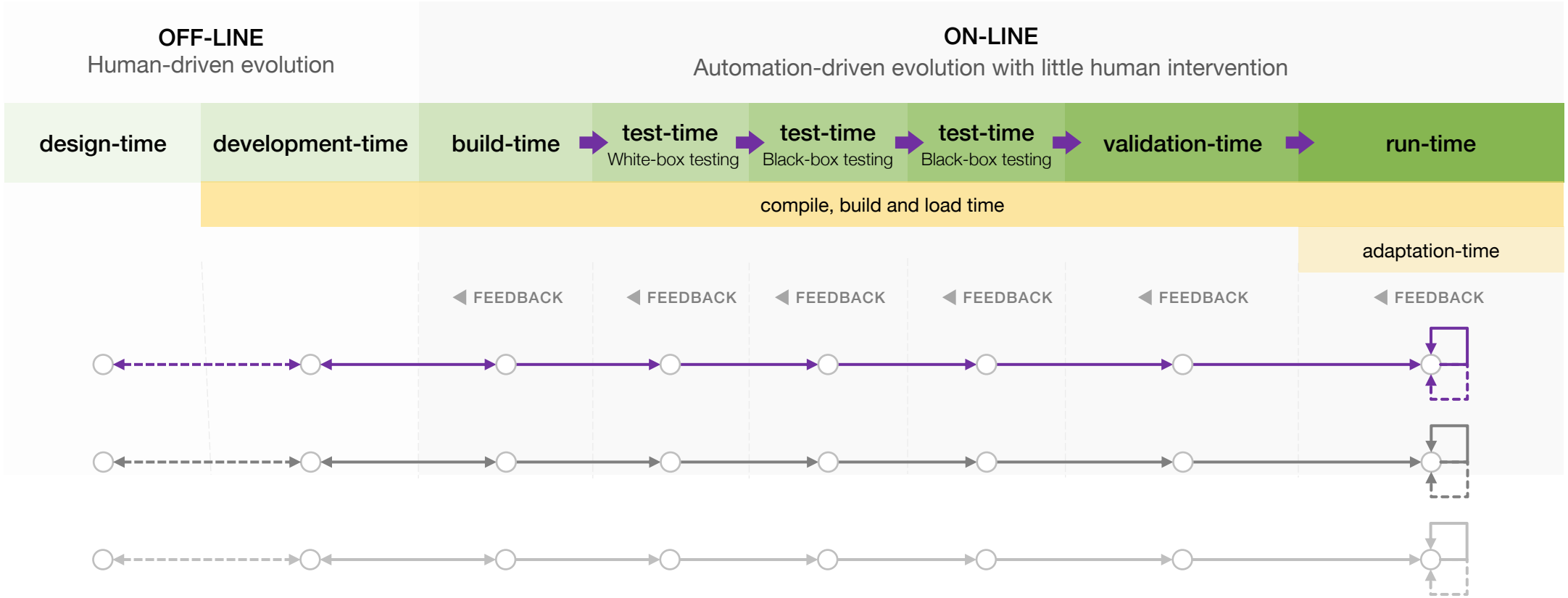


Legend

Environment | Manual change flow | Automated change flow | *-time Time-of-change timeline | Deployment-time | **QA**: Quality Assurance

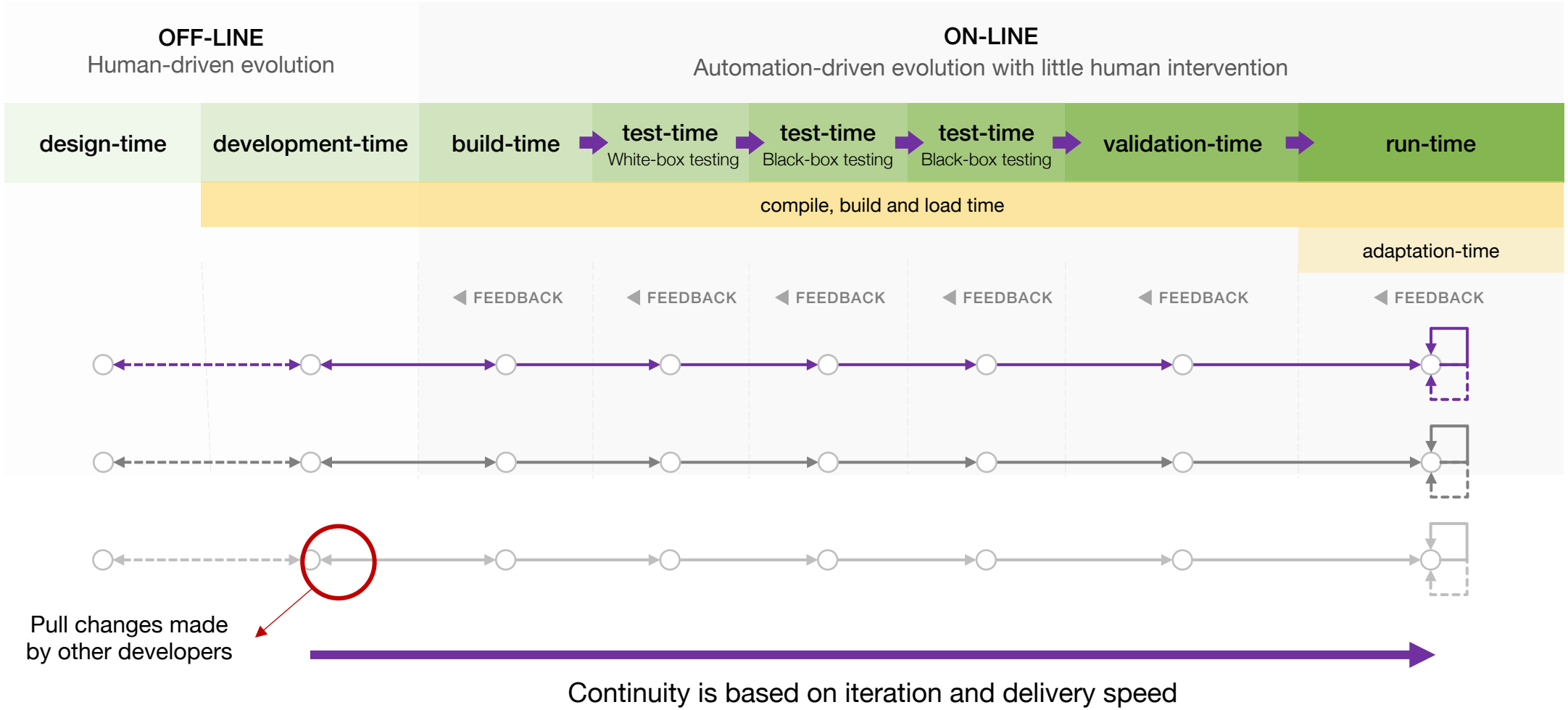
Off-line and On-line Reconceptualization

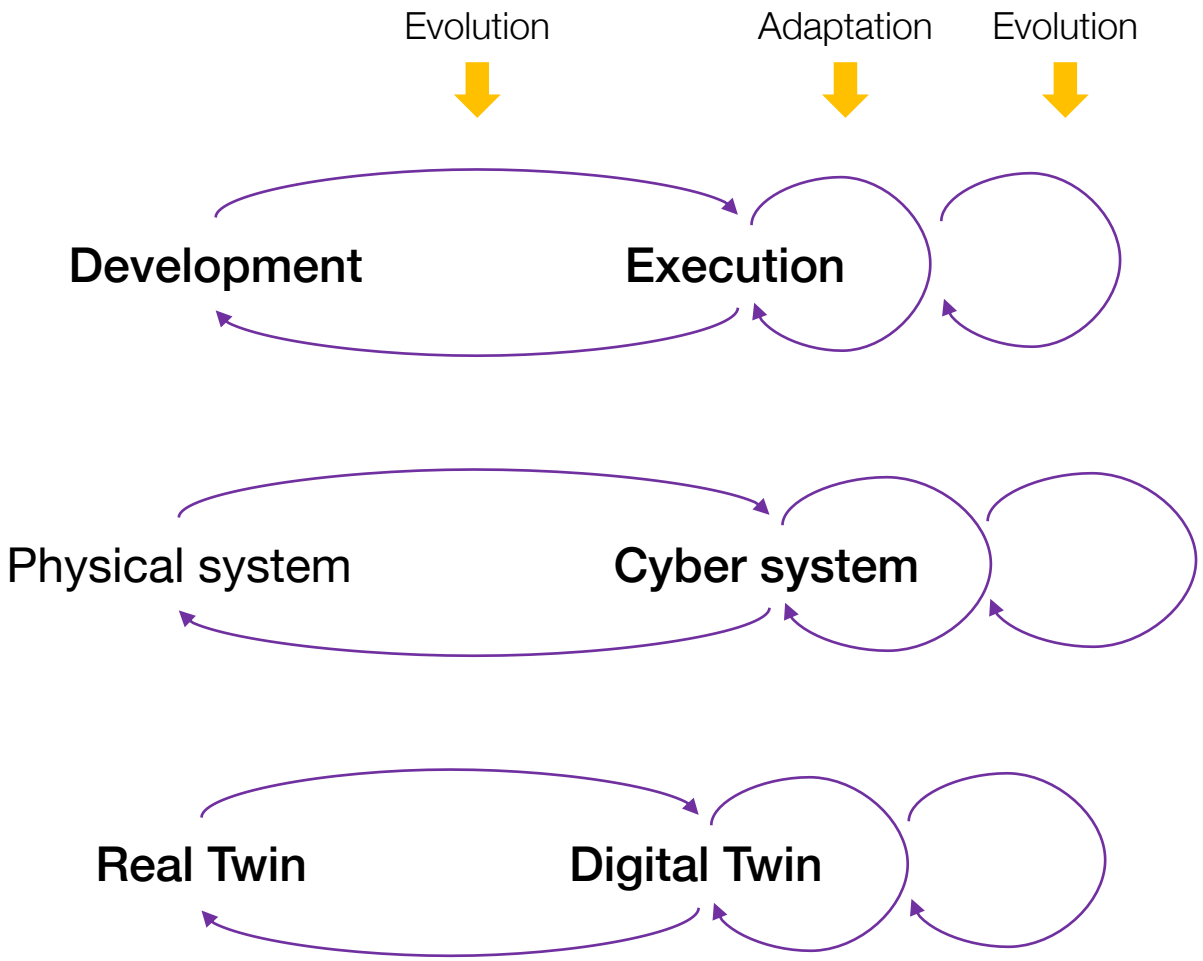
A holistic view of the time-of-change timeline today



Off-line and On-line Reconceptualization

A holistic view of the time-of-change timeline today





This evolution cycle manifests in various types of software-intensive systems

L. F. Rivera, M. Jiménez *et al*. **Designing Run-time Evolution for Dependable and Resilient Cyber-Physical Systems Using Digital Twins**, 2021 (in print)

Self-Evolution vs Self-Adaptation

Evolution from a Software Engineering perspective:

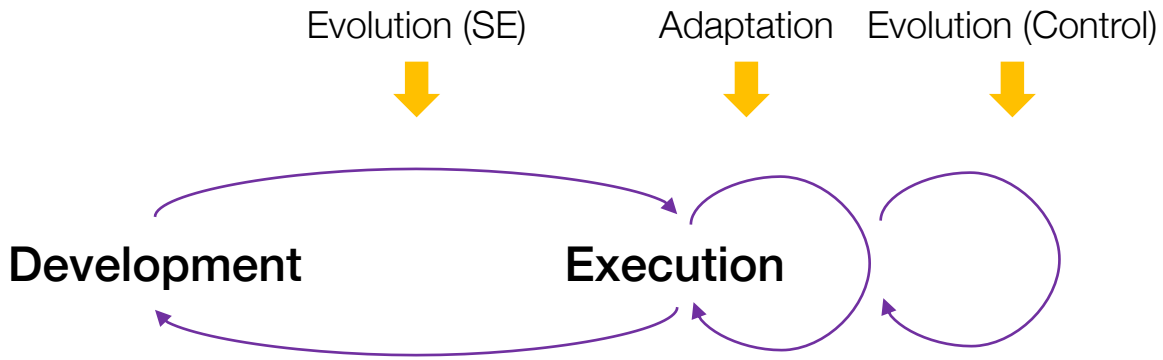
Evolution refers to changing the software system—producing a new version.

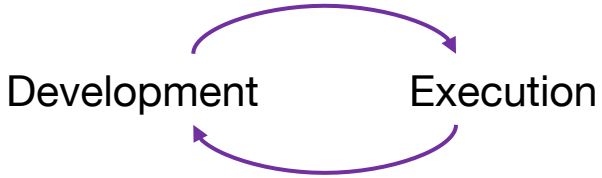
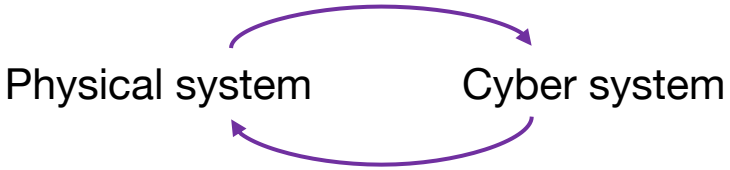
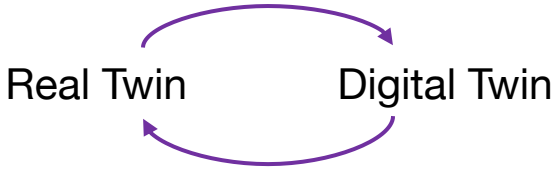
Evolution from a control perspective:

Evolution refers to changing the adaptation mechanism.

Adaptation:

Change the system's behavior.



	Self-Evolution (Engineering perspective)	**Self-Evolution** (Control perspective)	**Self-Adaptation**
	• Optimize deployment specifications in a repository	• Use grid search instead of random search when adaptation time is limited	• Increase / decrease the number of service instances
	• Improve the operational plan of a transportation system	• Use a Weibull distribution to describe the passenger's arrival times	• Decrease a bus line's interarrival time to reduce the passenger waiting time
	• Update an airplane's predictive maintenance schedule	• Use symbolic regression in addition to interpolation to mitigate precision errors	• Adapt the travel-time estimation based on weather conditions

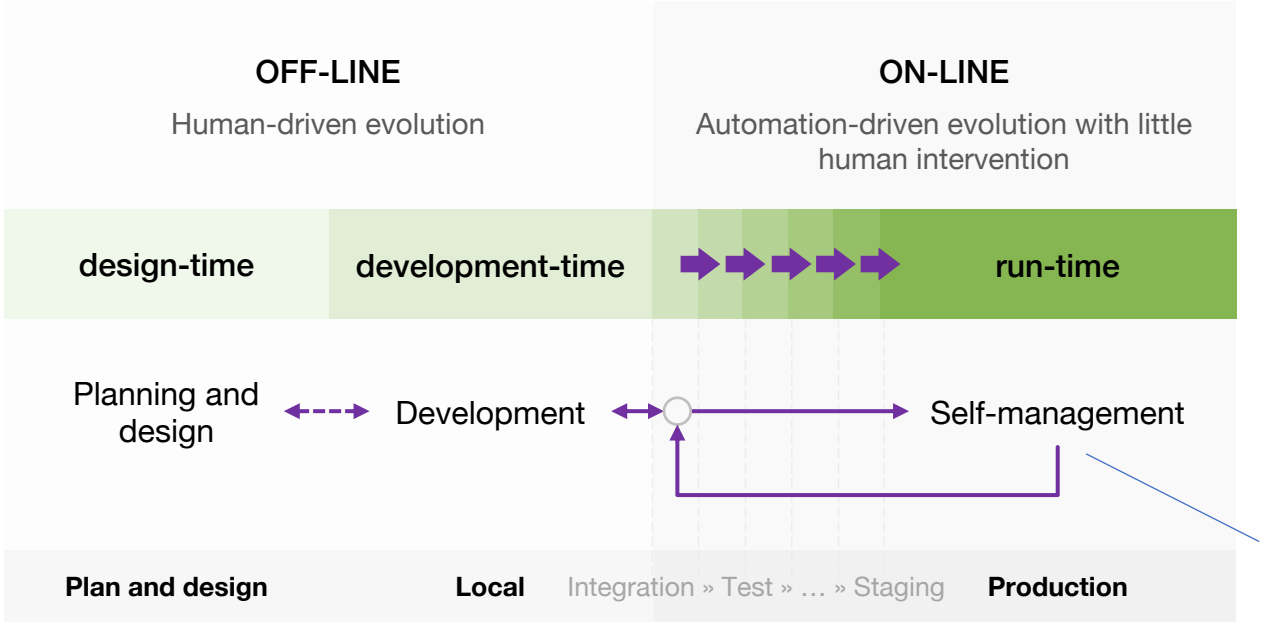
L. F. Rivera, M. Jiménez *et al*. **Designing Run-time Evolution for Dependable and Resilient Cyber-Physical Systems Using Digital Twins**, 2021 (in print)

General Evolution Outline: self-management (1)

Minor fixes and technical-debt corrections (Infrastructure-as-Code)

Examples:

- Automated program repair
- Technical debt
  - Infrastructure erosion
  - Configuration drift
  - Snowflake configuration
- Environment co-evolution



In addition to run-time management

M. Jiménez *et al*. DevOps Round-trip Engineering: Traceability From Dev to Ops and Back, 2018

M. Jiménez *et al*. Deployment Specification Challenges in the Context of Large Scale Systems, 2017

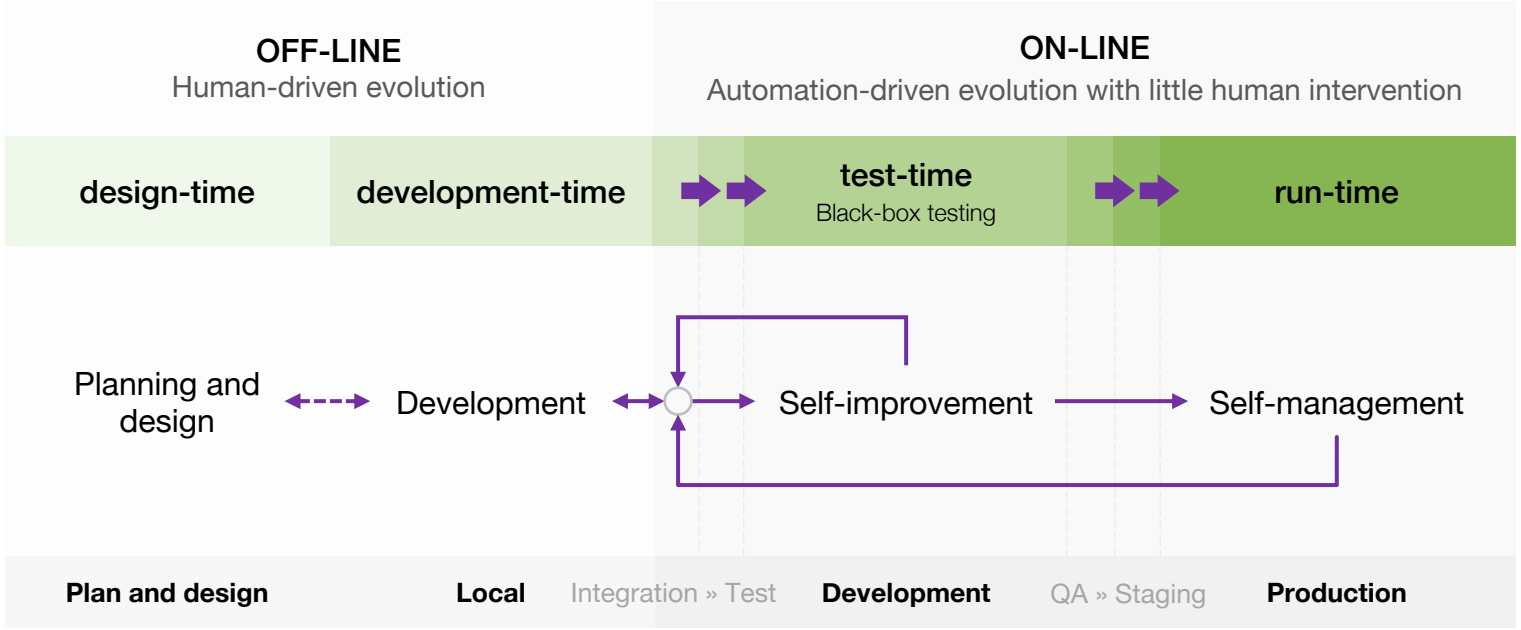
General Evolution Outline: self-management and self-improvement (2)

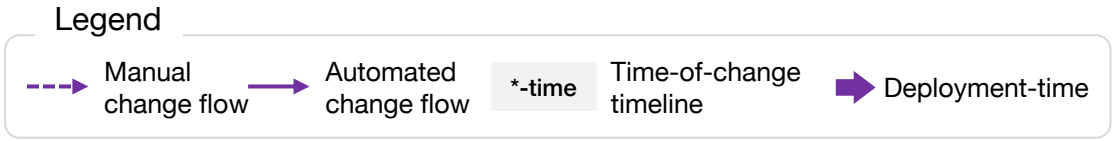
- **Minor fixes and technical-debt corrections (Infrastructure-as-Code)**
- **Experimentation-driven self-improvement during development**

Self-improvement is done in coordination with the release cycle

Examples:

- Continuous deployment design
- Continuous architecture design
- Software pattern selection





M. Jiménez *et al*. An Architectural Framework for Adaptive Quality-Driven Continuous Experimentation, 2019

M. Jiménez *et al*. DevOps' Shift-Left in Practice: An Industrial Case of Application, 2019

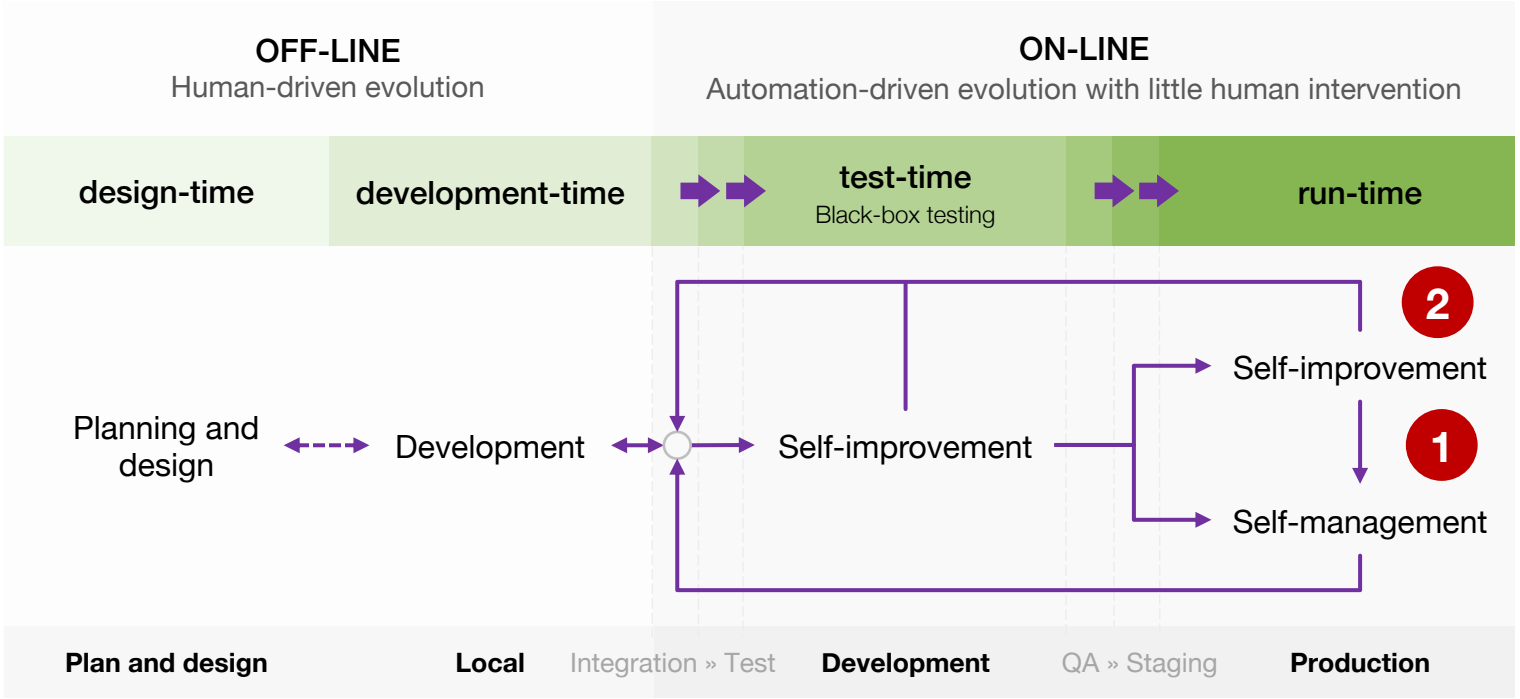
General Evolution Outline: self-management and self-improvement (3)

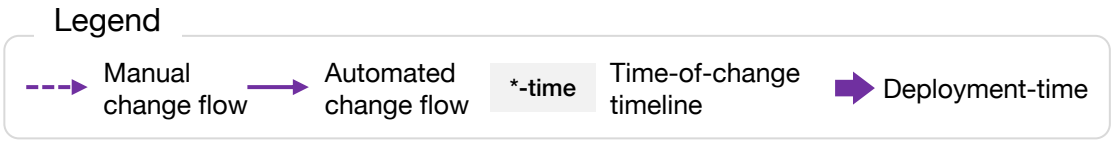
- **Minor fixes and technical-debt corrections (Infrastructure-as-Code)**
- **Experimentation-driven self-improvement during development**
- **Process-driven self-improvement at run-time**

2-way Continuous Delivery

Examples:

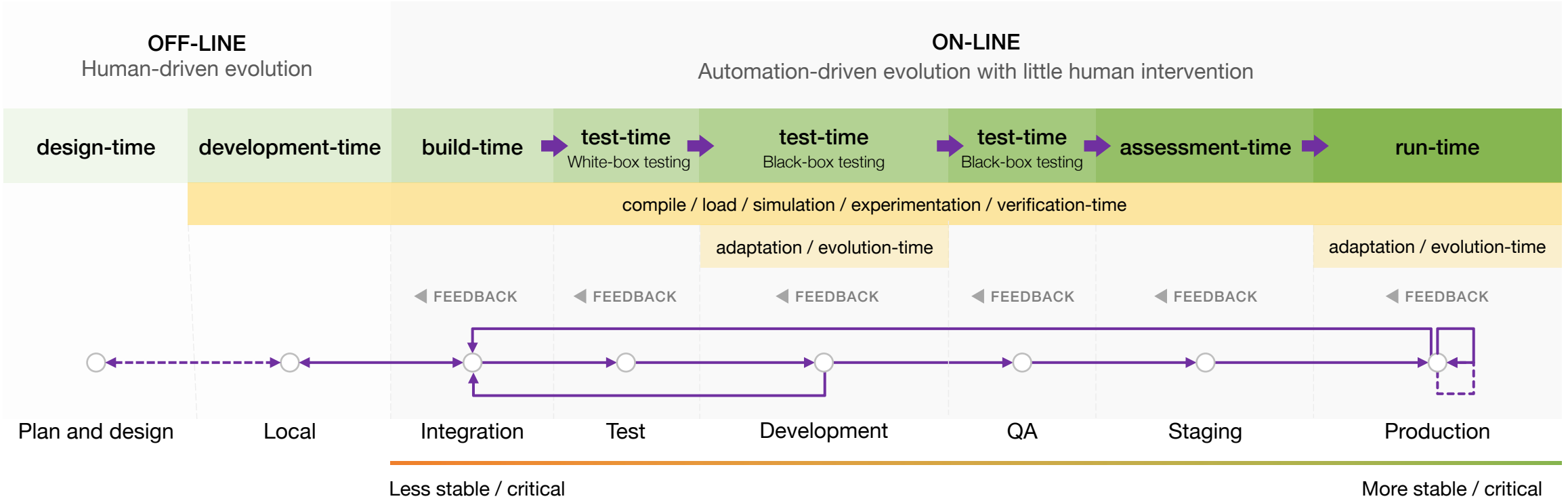
1. Adaptive control driven by process improvement
2. Improvement of development artifacts based on **emergent behavior**





Off-line and On-line Reconceptualization

A holistic view of the time-of-change timeline that we propose



Legend

Environment | Manual change flow | Automated change flow | *-time Time-of-change timeline | Deployment-time | **QA**: Quality Assurance

1

How to engineer self-managing systems?

2

Self-* operations based on strategies devised off-line

How to engineer change at run-time?

3

How can we better integrate off-line and on-line activities to develop continuous evolution?

Challenges

- **How to engineer change at run-time?**
  - Autonomic computing models are either too abstract or domain-specific
  - Most approaches from autonomic computing focus on knowledge management
  - The surprise factor is almost non-existent
  - How to deal with unforeseen changes? Still an open challenge

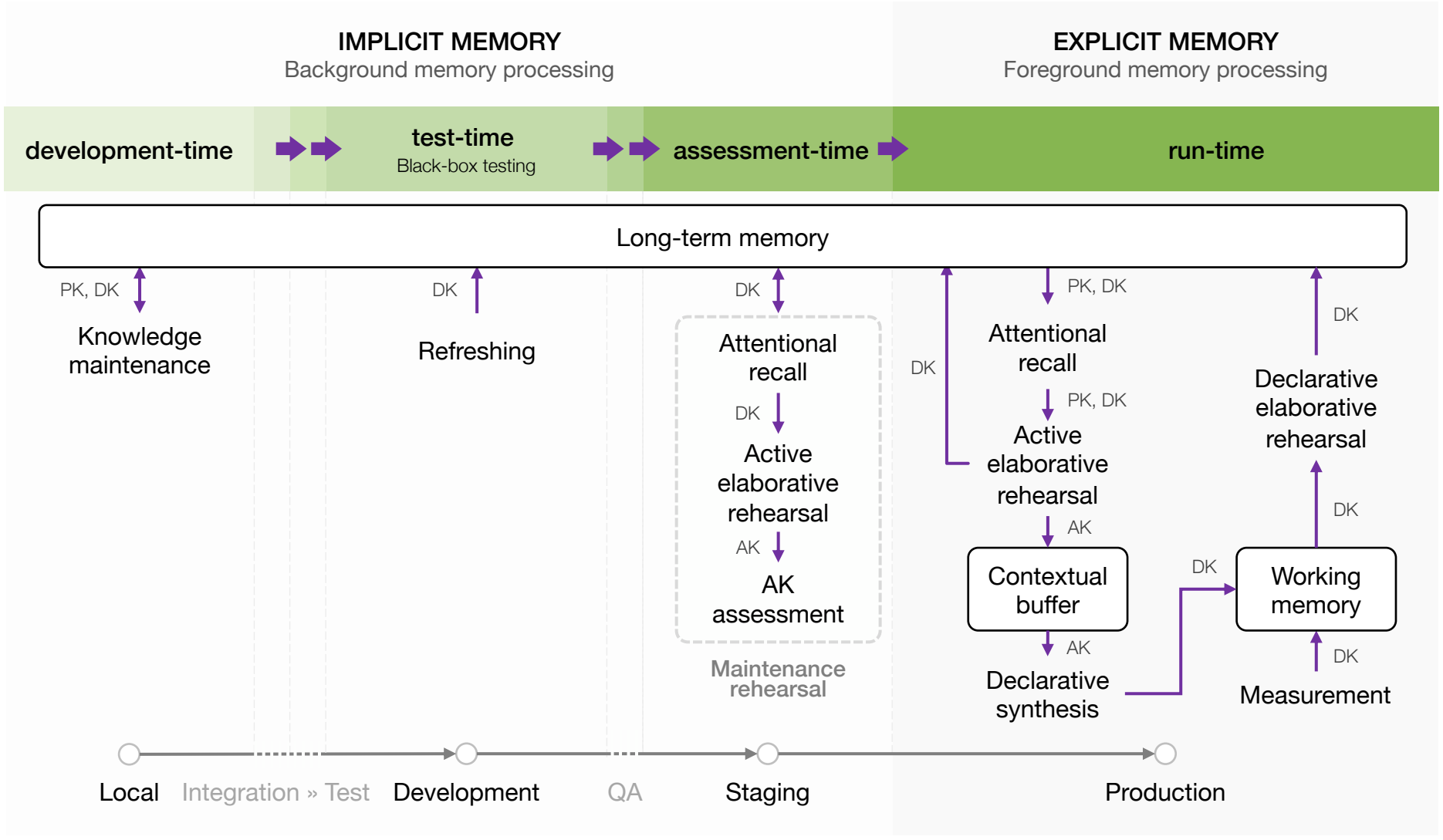
- **How can we better integrate off-line and on-line activities?**
  - Run-time activities reflect off-line processes, without the standardization
  - There's little progress on this (concrete contributions)

Our approach

- **Separate cognitive processes of knowledge development from reasoning**
  - Cognitive architecture for autonomic learning and decision making
  - Exploit the delivery pipeline infrastructure to learn faster
  - Model “good judgement” (common sense) decision making

- **Standardize on-line processes and integrate them with the SDLC**
  - Adapt processes from ISO 12207 to the run-time context
  - Create a life cycle model compatible with 2-way continuous delivery

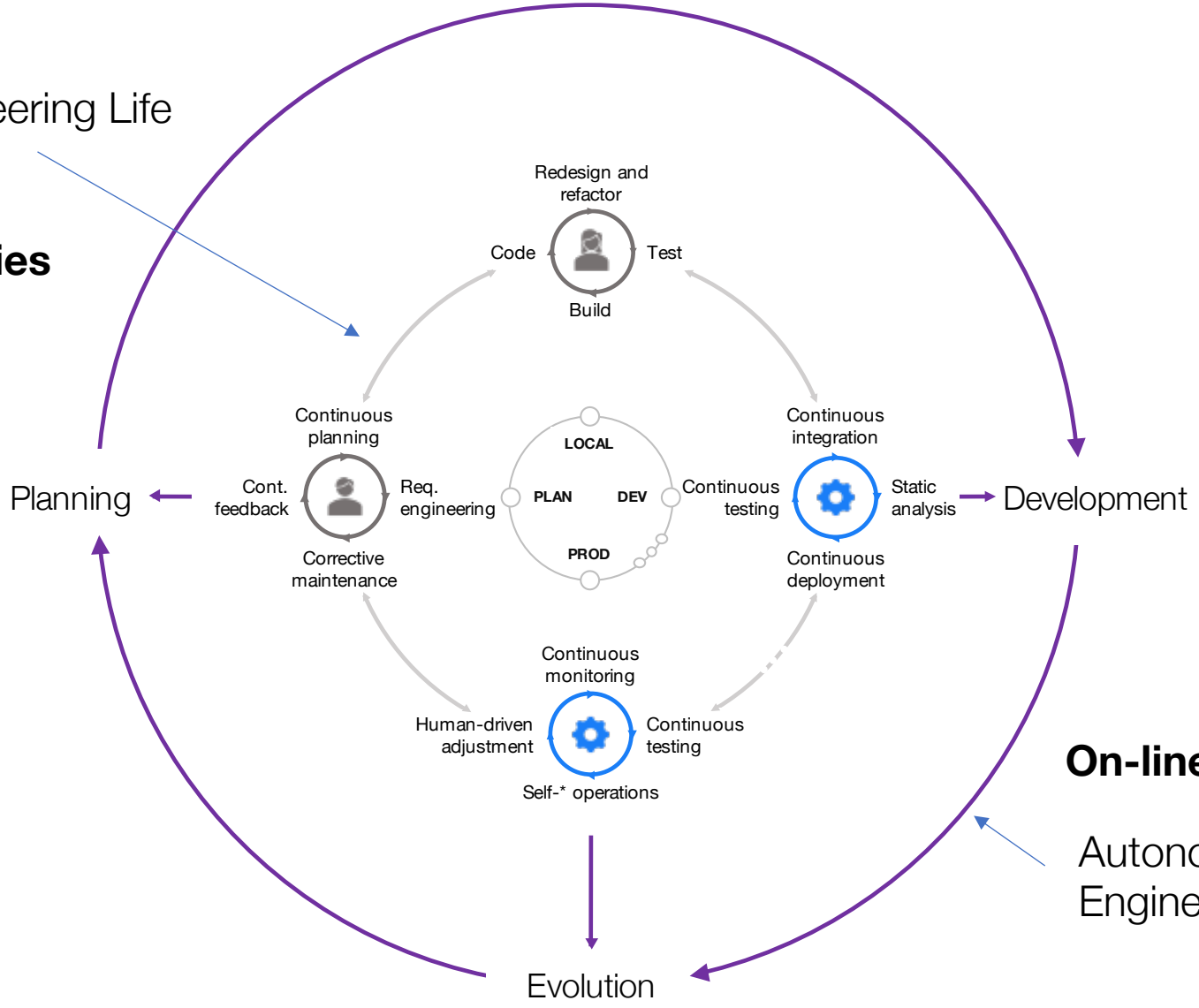
1. A Cognitive Architecture For Autonomic Learning and Decision Making



2. An Autonomic Software Engineering Life Cycle Model

Software Engineering Life Cycle Model

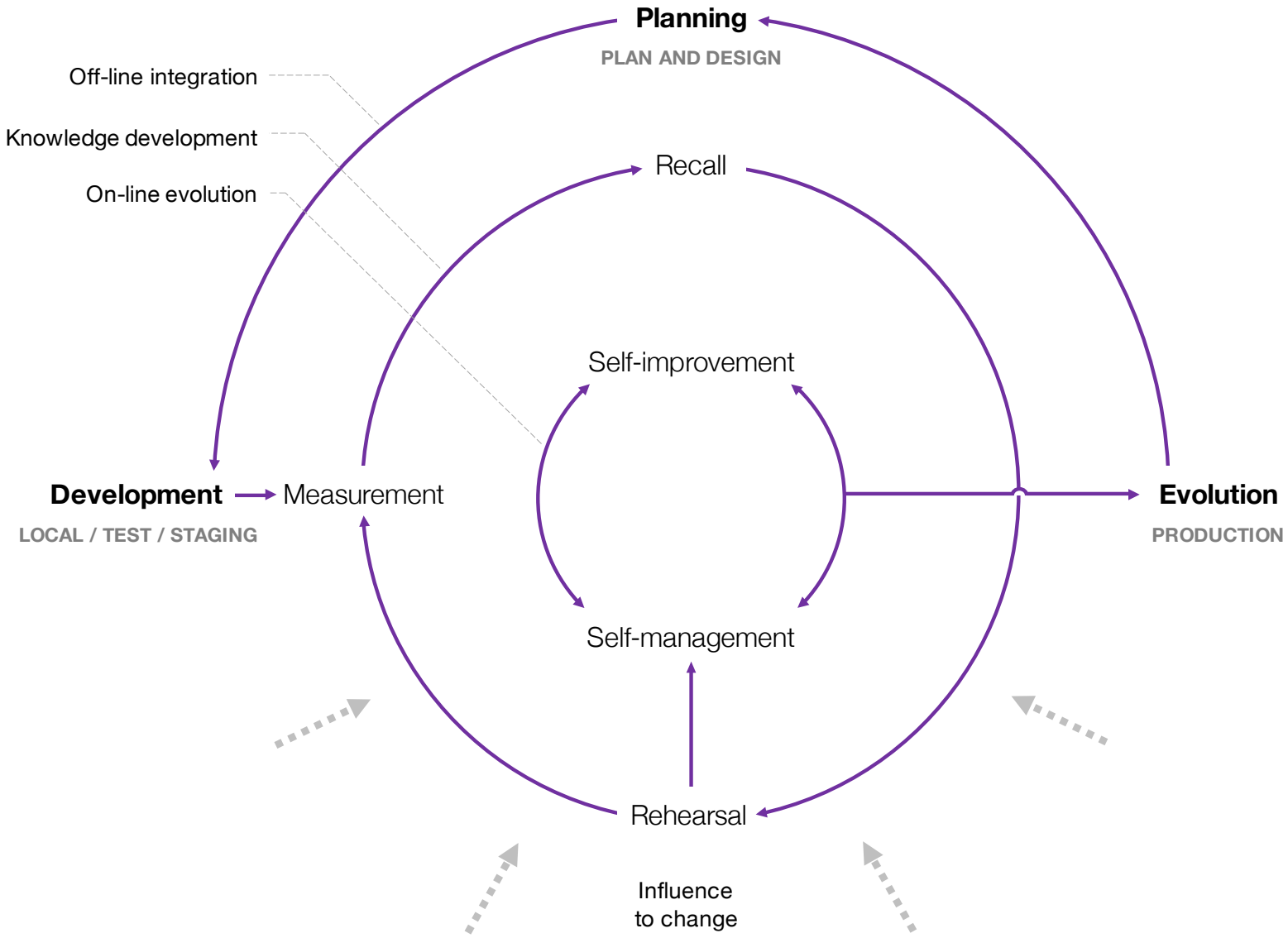
Off-line activities



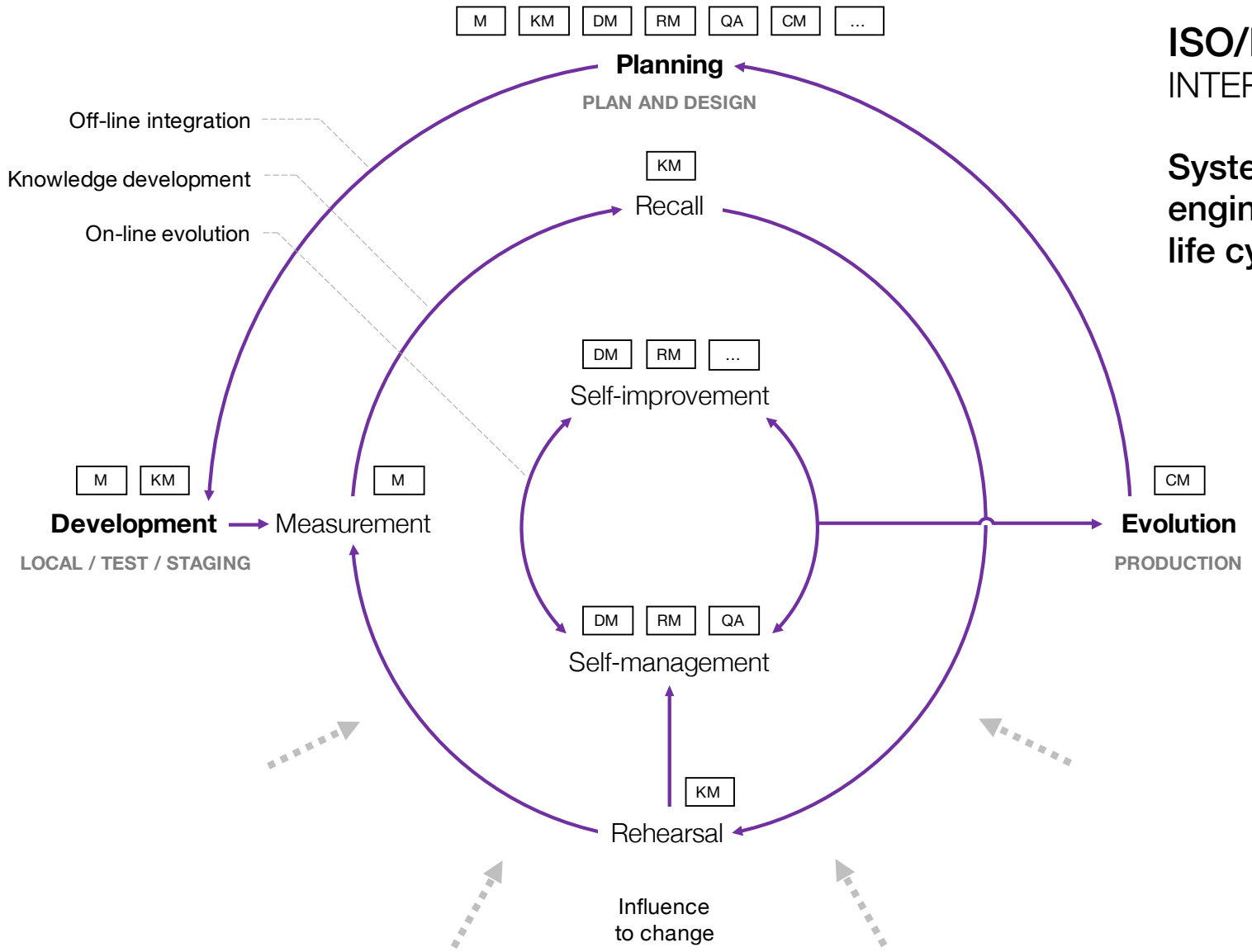
On-line activities

Autonomic Software Engineering Life Cycle Model

2. An Autonomic Software Engineering Life Cycle Model



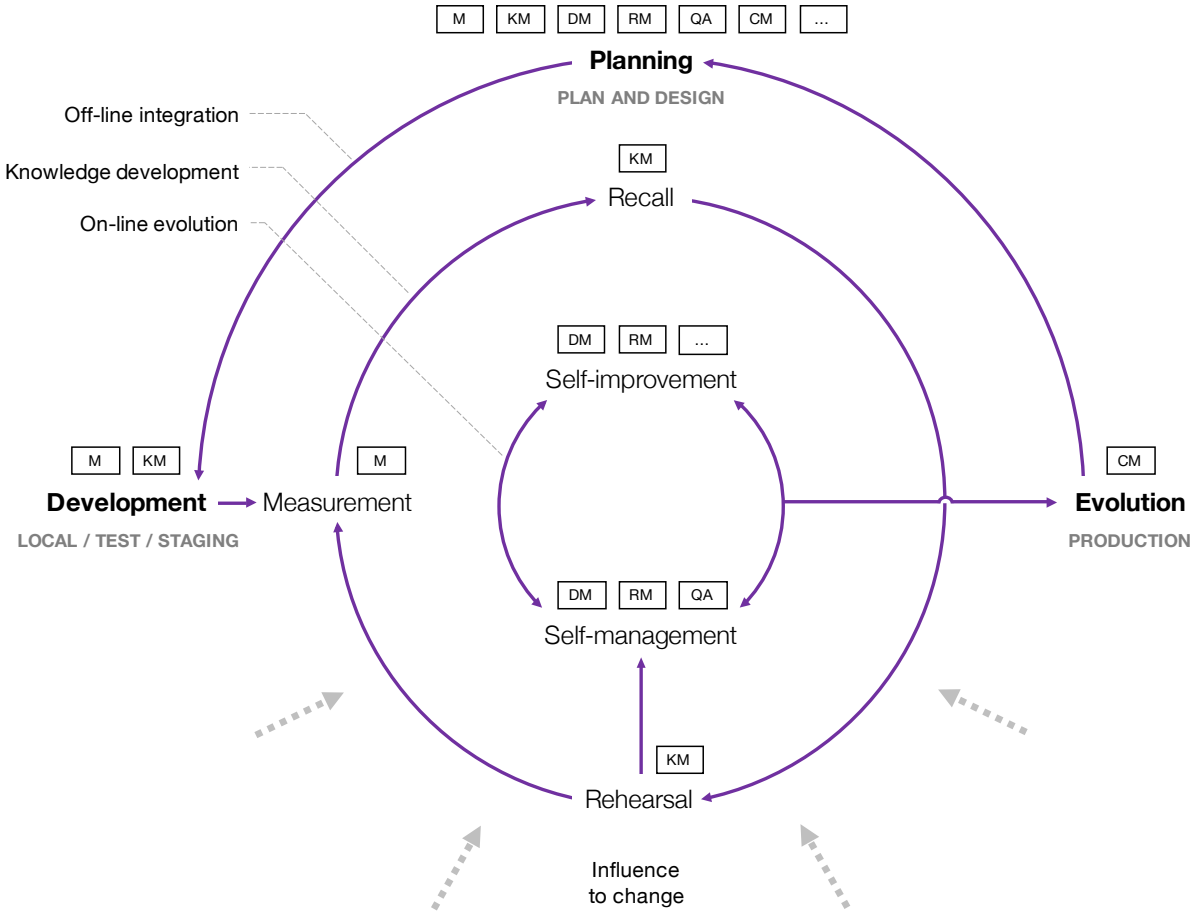
2. An Autonomic Software Engineering Life Cycle Model



ISO/IEC/IEEE 12207
INTERNATIONAL STANDARD

Systems and software engineering — Software life cycle processes

An Autonomic Software Engineering Life Cycle Model



Thank you!

Miguel Jiménez, Felipe Rivera,
Norha M. Villegas, Gabriel Tamura,
Hausi Müller

July 20th, 2021